

Εργαστήριο 9 - Άσκηση - Ανάλυση

Εκφώνηση: Δημιουργείτε ένα αντίγραφο της άσκησης της 8ης εργαστηριακής εβδομάδας. Κάντε τις απαραίτητες τροποποιήσεις ώστε να αντικαταστήσετε τα δύο `FileDialog` που χρησιμοποιήσατε για το Open και το Save με `JFileChooser`.

Προσοχή: Το Open `JFileChooser` ανοίγει με την εντολή

```
return Val = jOpenChooser.showOpenDialog(this);
```

ενώ το Save `JFileChooser` με την εντολή

```
return Val = jSaveChooser.showSaveDialog(this);
```

Η τιμή που επιστρέφει στην ακέραια μεταβλητή `return Val` μπορεί να είναι:

- ◆ `JFileChooser.APPROVE_OPTION`, αν πατηθεί το Open ή το Save
- ◆ `JFileChooser.CANCEL_OPTION`, αν πατηθεί το Cancel

Στη συνέχεια κάντε τις παρακάτω τροποποιήσεις:

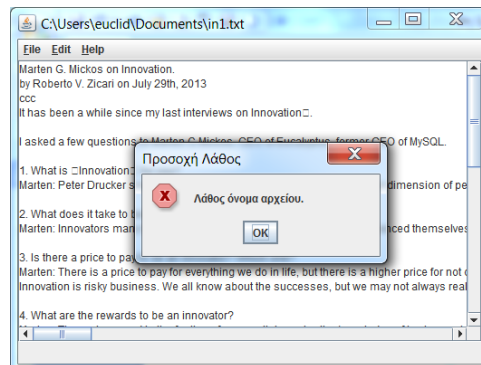
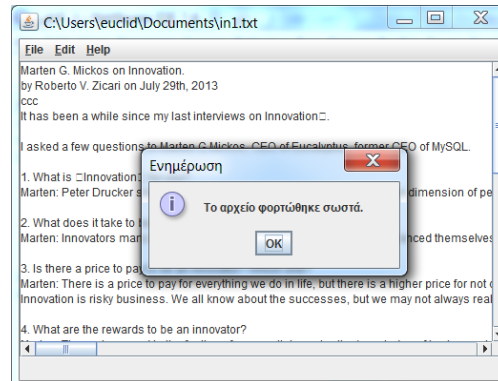
1. Δημιουργήστε δύο μεθόδους οι οποίες θα εμφανίζουν τα μηνύματα λάθους και τα μηνύματα ενημέρωσης του χρήστη με χρήση των

```
JOptionPane.showMessageDialog(null,  
    message,  
    "Προσοχή Λάθος",  
    JOptionPane.ERROR_MESSAGE);
```

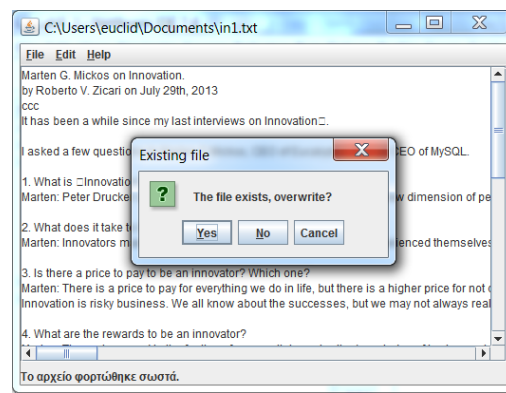
```
JOptionPane.showMessageDialog(null,  
    message,  
    "Ενημέρωση",  
    JOptionPane.INFORMATION_MESSAGE);
```

Σημείωση: η παράμετρος `message` είναι το κείμενο που θέλετε να εμφανίζετε στο παράθυρο, η επόμενη παράμετρος είναι ο τίτλος του παραθύρου και η τελευταία παράμετρος είναι η εικόνα που εμφανίζετε στο παράθυρο.

2. Αντικαταστήστε όλες τις εμφανίσεις μηνυμάτων στο πρόγραμμα σας με τις δικές σας μεθόδους ενημέρωσης. Στις επόμενες δύο εικόνες φαίνονται αντίστοιχα παραδείγματα.



3. Χρησιμοποιήστε ένα JOptionPane Swing Window στη λειτουργία «Save As» για τη διαχείριση της αντικατάστασης του περιεχομένου ενός αρχείου που υπάρχει, όπως φαίνεται στην παρακάτω εικόνα.



Σημείωση 1: Για να κάνετε τον έλεγχο αν ένα αρχείο υπάρχει θα χρειαστεί να δημιουργήσετε ένα αντικείμενο της τάξης *File*. Έτσι, αν δηλώσετε ένα αντικείμενο *fileobject* της τάξης *File* τότε με τις παρακάτω εντολές μπορείτε να κάνετε τον έλεγχο αν υπάρχει το αρχείο.

```
fileobject = jSaveChooser.getSelectedFile();  
If(fileobject.exists()) ...
```

Σημείωση 2: Το JOptionPane θα το χρησιμοποιήσετε με τη μέθοδο

```
int result = JOptionPane.showConfirmDialog(this, "The file exists, overwrite?",
    "Existing file", JOptionPane.YES_NO_CANCEL_OPTION);
```

Η τιμή που επιστρέφει στην ακέραια μεταβλητή *result* μπορεί να είναι:

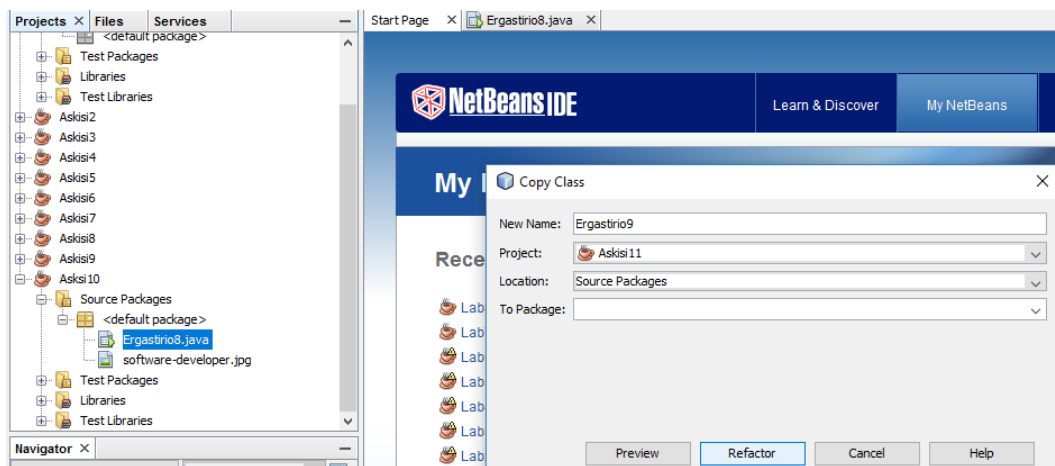
- ◆ JOptionPane.YES_OPTION, αν πατηθεί το Yes,
- ◆ JOptionPane.NO_OPTION, αν πατηθεί το No,
- ◆ JOptionPane.CLOSED_OPTION, αν κλείσουμε το παράθυρο,
- ◆ JOptionPane.CANCEL_OPTION, αν πατηθεί το Cancel.

Ανάλυση και εξήγηση λύσης

Βήμα 1: Δημιουργούμε ένα νέο project στο NetBeans με όνομα Askisi11.

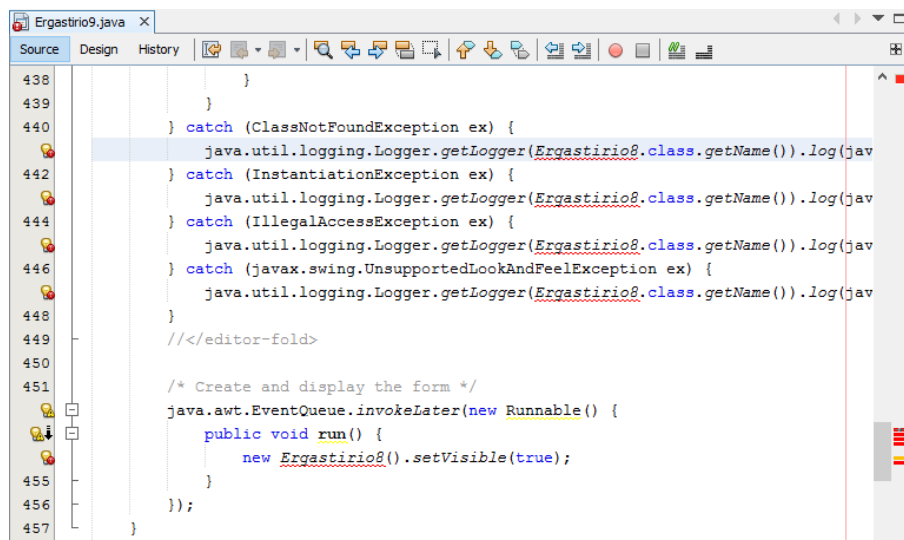
Βήμα 2: Αντιγράφουμε το Java Swing αρχείο «Ergastirio8.java» από το project Askisi10.

Η αντιγραφή γίνεται με Refactor copy επιλέγοντας το αρχείο «Ergastirio8.java», όπως φαίνεται στην Εικόνα 1.



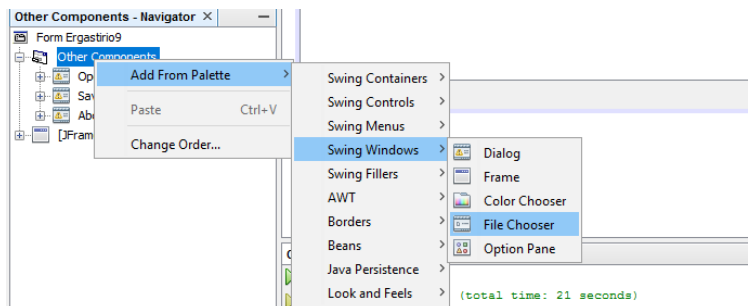
Εικόνα 1: Αντιγραφή αρχείου από άλλο project

Βήμα 3: Ανοίξετε το Ergastirio9. Αλλάξτε στον κώδικα κάθε αναφορά του «Ergastirio8» (Εικόνα 2) σε «Ergastirio 9». Αλλάξτε την ιδιότητα title του JFrame σε «My Editor».



Εικόνα 2: Η αναφορά στο Ergastirio8 σημειώνετε με λάθος.

Βήμα 4: Προσθέστε από το Navigator στα «Other Components» (Εικόνα 3) δύο JFileChooser Window, με ονόματα μεταβλητών OpenFileChooser και SaveFileChooser.



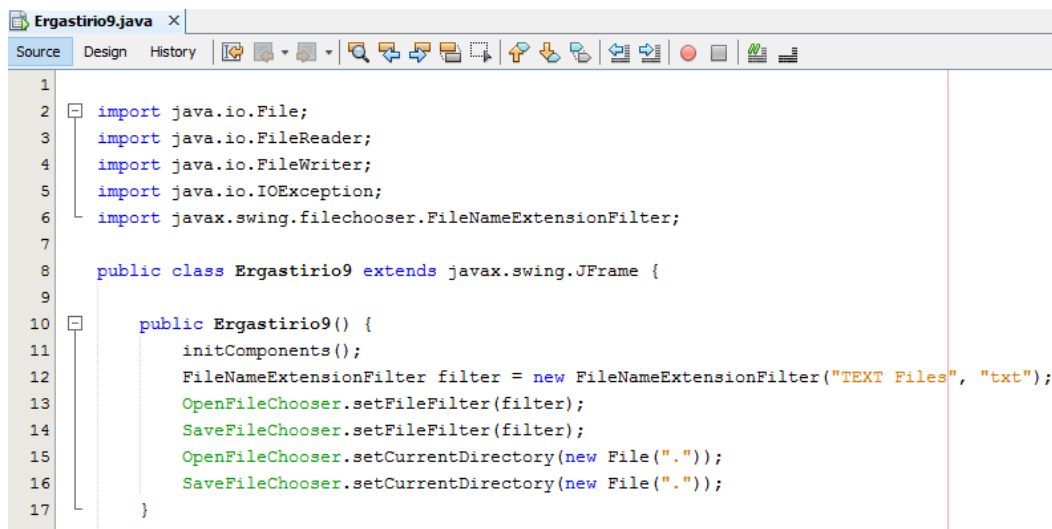
Εικόνα 3: Εισαγωγή File Chooser στα other Components

Βήμα 5: Στο δομητή της τάξης, μετά την initComponents(), προσθέτουμε τον κώδικα που φαίνεται στην Εικόνα 4, στις γραμμές 12-16.

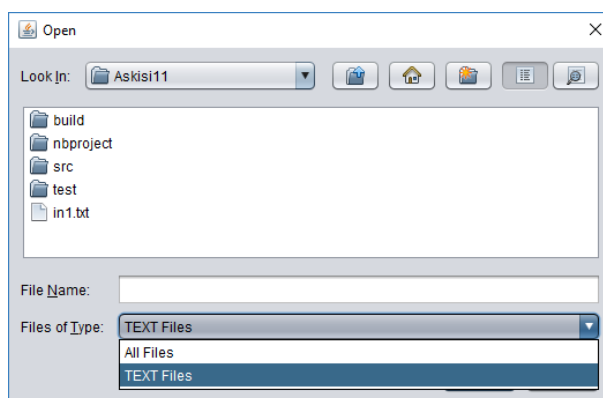
Στη γραμμή 12, ορίζουμε ένα αντικείμενο της τάξης FileNameExtensionFilter, το οποίο θα περιέχει το φίλτρο ("TEXT Files", "txt"). Αυτό εμφανίζει στο File Chooser το κείμενο «TEXT files» στα «Files of Types» (Εικόνα 5), και επίσης εμφανίζει μόνο τα αρχεία με κατάληξη «txt» (Εικόνα 5).

Στις γραμμές 13 και 14, εφαρμόζουμε το φίλτρο στα δύο File Chooser.

Στις γραμμές 15 και 16, ορίζουμε ως current directory το «.» για τα δύο File Chooser. Δηλαδή, όταν ανοίγει το File Chooser θα εμφανίζει τον κατάλογο του project Askisi11.



Εικόνα 4: Ορισμός φίλτρων για τα File Chooser



Εικόνα 5: Το OpenFileChooser

Βήμα 6: Αλλάζουμε τον κώδικα του γεγονότος Action Performed της επιλογής Open του μενού File με τον κώδικα που φαίνεται στην Εικόνα 6. Αναλυτικά:

Γραμμή 226: Δήλωση αντικείμενου της τάξης FileReader για ανάγνωση αρχείου.

Γραμμή 227: Ανοίγει το OpenFileChooser με τη μέθοδο showOpenDialog, όπου το κουμπί έχει το κείμενο «Open». Στην ακέραιο μεταβλητή returnVal αποθηκεύεται ένας ακέραιο, όταν κλείσει ο χρήστης το File Chooser, ανάλογα με τον τρόπο που θα το κλείσει. Δηλαδή, επιστρέφει το «JFileChooser.APPROVE_OPTION» όταν πατηθεί το κουμπί «Open» και το «JFileChooser.CANCEL_OPTION» όταν πατηθεί το κουμπί «Cancel» ή κλείσει το παράθυρο.

Γραμμή 228: Στο αντικείμενο fileobject της τάξης File αποθηκεύεται το όνομα του επιλεγμένου αρχείου.

Γραμμή 229: Γίνεται έλεγχος αν το fileobject έχει τιμή και αν έχει πατηθεί το κουμπί «Open».

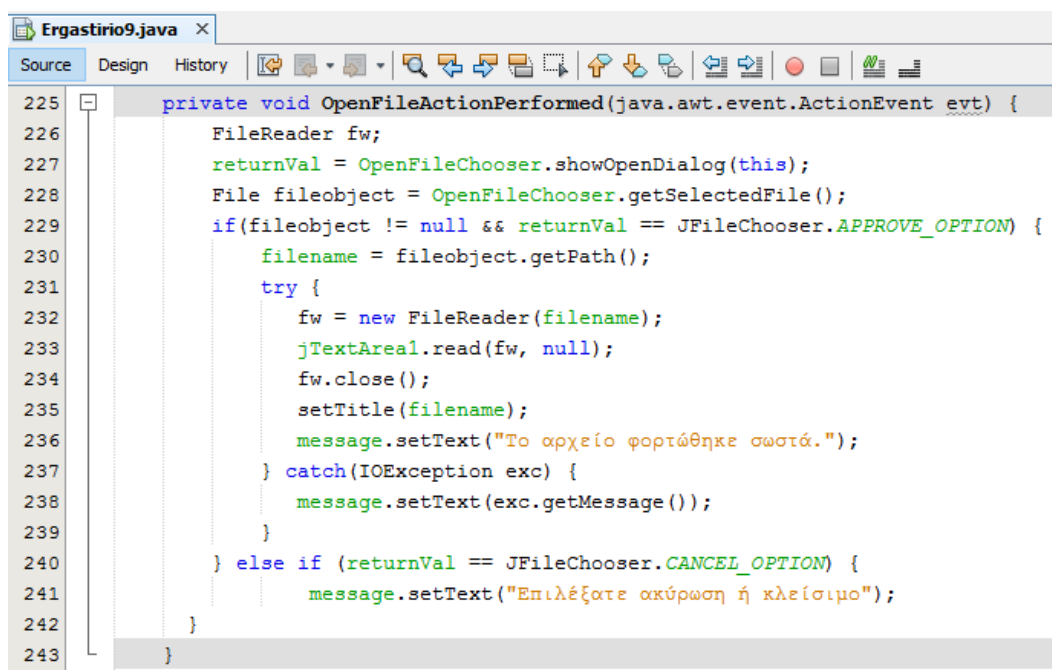
Γραμμή 230: Αν ναι αποθηκεύεται στην αλφαριθμητική μεταβλητή filename η διαδρομή και το όνομα του αρχείου που επιλέχθηκε.

Γραμμή 231 - 239: Μέσα σε try-catch (IOException) γίνεται ανάγνωση του αρχείου και εμφάνισή του στο TextArea.

Γραμμή 240 - 242: Αν το File Chooser έχει κλείσει χωρίς να πατηθεί το κουμπί «Open» (αλλά το Cancel ή το Close). Στην ετικέτα message εμφανίζεται κατάλληλο μήνυμα.

Επίσης, ορίζουμε δύο global μεταβλητές, την αλφαριθμητική filename και τον ακέραιο returnVal.

Τέλος, διαγράφουμε το OpenFileDialog από το «Other Components» και διαγράφονται και όλα τα γεγονότα που έχουν οριστεί σε συστατικά αυτού.



```

225 private void OpenFileActionPerformed(java.awt.event.ActionEvent evt) {
226     FileReader fw;
227     returnVal = OpenFileChooser.showOpenDialog(this);
228     File fileobject = OpenFileChooser.getSelectedFile();
229     if(fileobject != null && returnVal == JFileChooser.APPROVE_OPTION) {
230         filename = fileobject.getPath();
231         try {
232             fw = new FileReader(filename);
233             jTextArea1.read(fw, null);
234             fw.close();
235             setTitle(filename);
236             message.setText("Το αρχείο φορτώθηκε σωστά.");
237         } catch(IOException exc) {
238             message.setText(exc.getMessage());
239         }
240     } else if (returnVal == JFileChooser.CANCEL_OPTION) {
241         message.setText("Επιλέξατε ακύρωση ή κλείσιμο");
242     }
243 }

```

Εικόνα 6: Ο κώδικας του OpenFileChooser

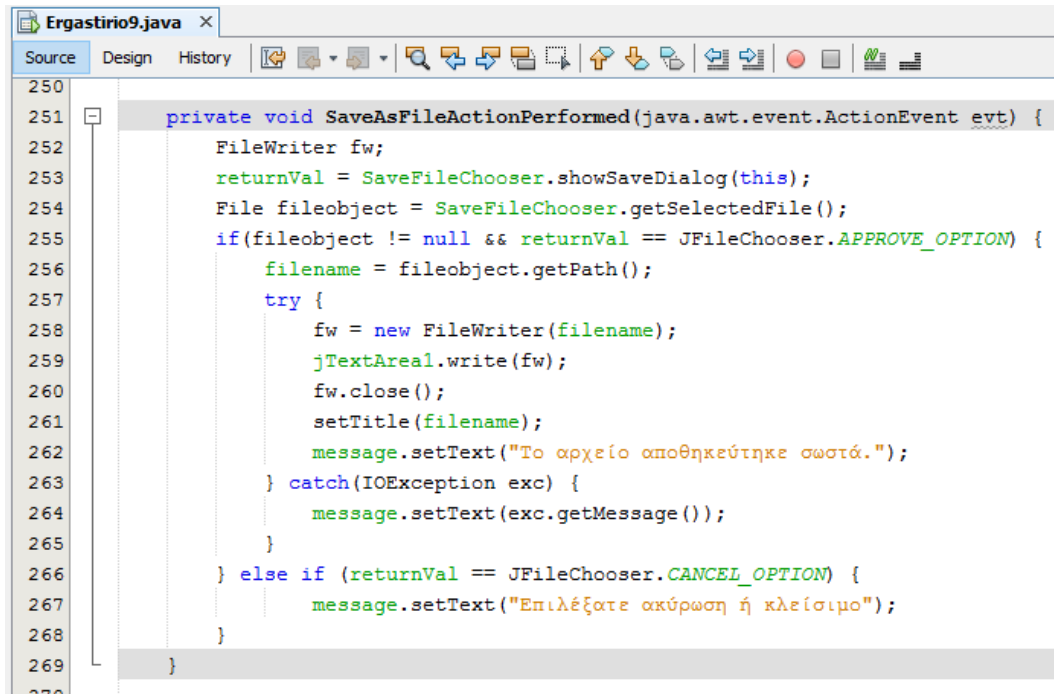
Βήμα 7: Αλλάζουμε τον κώδικα του γεγονότος Action Performed της επιλογής «Save As» του μενού File με τον κώδικα που φαίνεται στην Εικόνα 7. Ο κώδικας είναι παρόμοιος με τον κώδικα του Open, με μικρές διαφορές. Συγκεκριμένα:

Στη γραμμή 252 γίνεται δήλωση αντικείμενου της τάξης FileWriter για αποθήκευση σε αρχείο.

Στη γραμμή 253, το SaveFileChooser θα ανοίξει με τη μέθοδο showSaveDialog, όπου το κουμπί θα έχει το κείμενο «Save».

Στη γραμμή 259, γίνεται αποθήκευση του περιεχομένου του TextArea σε αρχείο κειμένου με τη μέθοδο write.

Τέλος, διαγράφουμε το SaveDialog από το «Other Components» και διαγράφονται και όλα τα γεγονότα που έχουν οριστεί σε συστατικά αυτού.



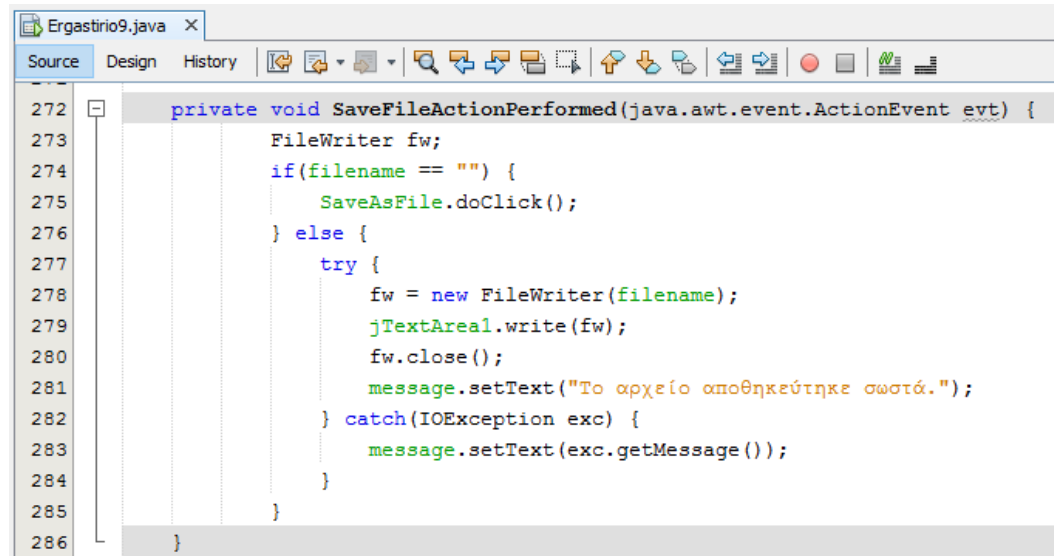
```

250
251 private void SaveAsFileActionPerformed(java.awt.event.ActionEvent evt) {
252     FileWriter fw;
253     returnVal = SaveFileChooser.showSaveDialog(this);
254     File fileobject = SaveFileChooser.getSelectedFile();
255     if(fileobject != null && returnVal == JFileChooser.APPROVE_OPTION) {
256         filename = fileobject.getPath();
257         try {
258             fw = new FileWriter(filename);
259             jTextArea1.write(fw);
260             fw.close();
261             setTitle(filename);
262             message.setText("Το αρχείο αποθηκεύτηκε σωστά.");
263         } catch(IOException exc) {
264             message.setText(exc.getMessage());
265         }
266     } else if (returnVal == JFileChooser.CANCEL_OPTION) {
267         message.setText("Επιλέξατε ακύρωση ή κλείσιμο");
268     }
269 }
270

```

Εικόνα 7: Ο κώδικας του Save As

Ο κώδικας του γεγονότος Action Performed της επιλογής «Save» του μενού File παραμένει ίδιος με τη προηγούμενη άσκηση όπως φαίνεται Εικόνα 8.



```

272 private void SaveFileActionPerformed(java.awt.event.ActionEvent evt) {
273     FileWriter fw;
274     if(filename == "") {
275         SaveAsFile.doClick();
276     } else {
277         try {
278             fw = new FileWriter(filename);
279             jTextArea1.write(fw);
280             fw.close();
281             message.setText("Το αρχείο αποθηκεύτηκε σωστά.");
282         } catch(IOException exc) {
283             message.setText(exc.getMessage());
284         }
285     }
286 }

```

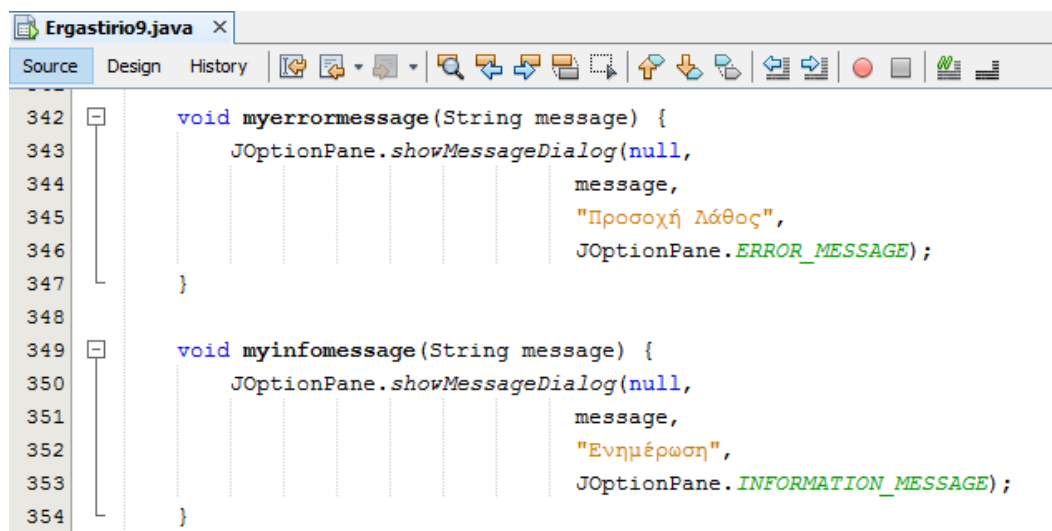
Εικόνα 8: Ο κώδικας του File Save

Βήμα 8: Σε αυτό το βήμα θα δημιουργήσουμε δύο void δικές μας μεθόδους, όπως φαίνεται στην Εικόνα 9, για να εμφανίζουμε τα μηνύματα σε παράθυρα.

Η πρώτη μέθοδος, η myErrorMessage, παίρνει ως παράμετρο το μήνυμα λάθους και το παρουσιάζει σε ένα παράθυρο με χρήση της τάξης JOptionPane και της μεθόδου

showMessageDialog. Η JOptionPane δέχεται τέσσερις παραμέτρους. Η πρώτη δείχνει τον υποδοχέα (container) που ανήκει το παράθυρο JOptionPane. Με την τιμή null δείχνει ότι ανήκει στο JFrame. Η δεύτερη παράμετρος δείχνει το μήνυμα λάθους στον καμβά του παραθύρου. Η τρίτη παράμετρος είναι ο τίτλος του παραθύρου. Η τέταρτη παράμετρος είναι η εικόνα που θα εμφανιστεί στο παράθυρο.

Η δεύτερη μέθοδος, η myinfomessage, εμφανίζει μηνύματα ενημέρωσης στον χρήστη. Και αυτή η μέθοδος στηρίζεται πάνω στην JOptionPane και τη μέθοδο showMessageDialog, όπου οι παράμετροι της δηλώνονται παρόμοια.



Εικόνα 9: Ο κώδικας των δύο μεθόδων εμφάνισης μηνυμάτων και λαθών

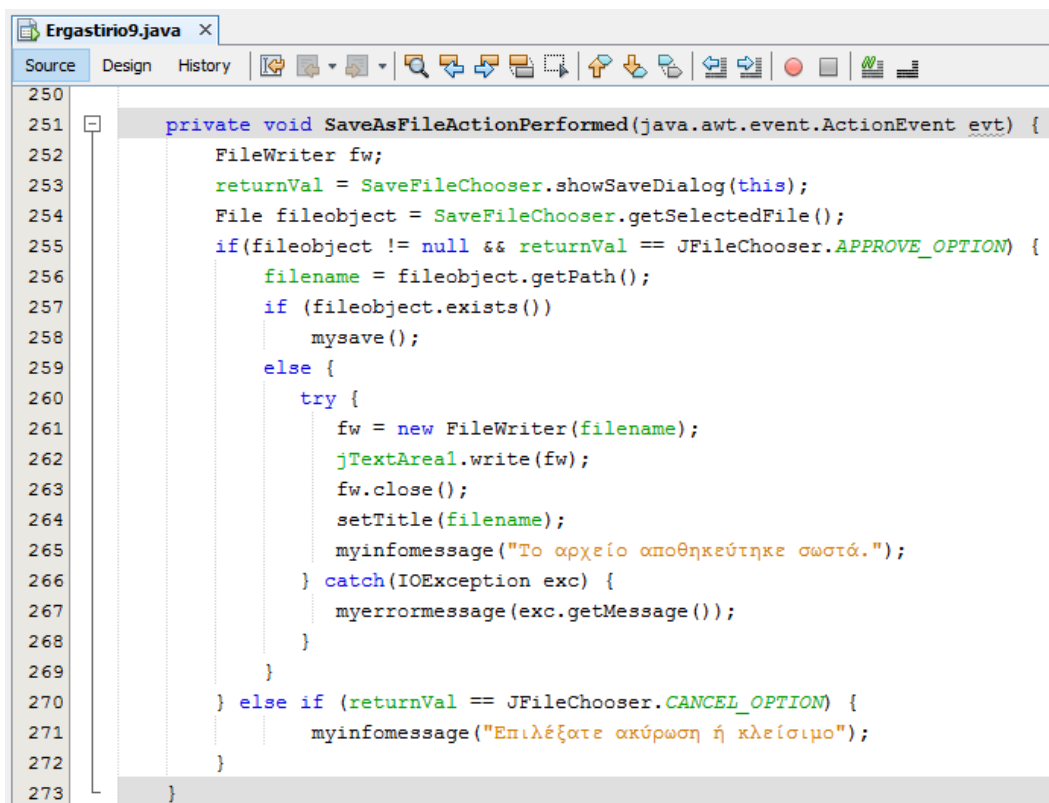
Βήμα 9: Σε αυτό το βήμα θα χρησιμοποιούμε τις παραπάνω δύο μεθόδους για την εμφάνιση των λαθών και των μηνυμάτων.

Έτσι, αλλάζουμε τις εντολές στο τμήμα της catch στα try-catch, με την myerrorMessage(exc.getMessage()).

Επίσης, αλλάζουμε όλες τις υπόλοιπες message.setText(μήνυμα) με την εντολή myinfomessage(μήνυμα).

Βήμα 10: Στη λειτουργία «Save As» υπάρχει η περίπτωση να ζητήσει ο χρήστης να γίνει αντικατάσταση ενός υπάρχοντος αρχείου με νέο περιεχόμενο. Η αναμενόμενη αντίδραση του κειμενογράφου είναι η εμφάνιση ενός παραθύρου, το οποίο ρωτάει τον χρήστη αν θέλει να κάνει την αντικατάσταση ή όχι. Αυτό ακριβώς υλοποιείται σε αυτό το βήμα.

Έτσι, στην Εικόνα 10, φαίνεται ο κώδικας της «Save As» τροποποιημένος για να διαχειριστεί τη λειτουργία της αντικατάστασης. Αυτό φαίνεται στις γραμμές 257 – 259. Στην γραμμή 257 γίνεται έλεγχος αν υπάρχει το αρχείο και στη 258 καλείται μία δική μας μέθοδος (η mysave) στην οποία θα διαχειριστούμε την αντικατάσταση του αρχείου.



Εικόνα 10: Η Save As με διαχείριση αντικατάστασης αρχείου

Στην Εικόνα 11, φαίνεται ο κώδικας της mysave. Αναλυτικά:

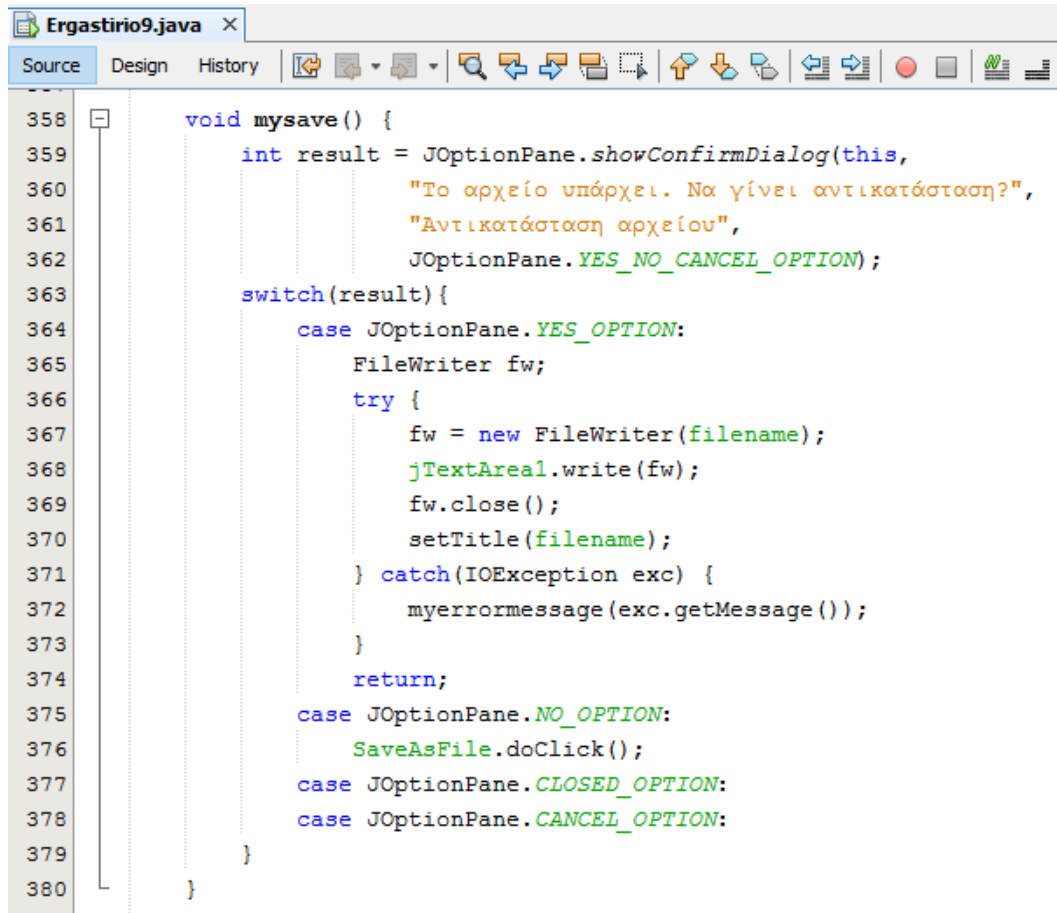
Γραμμή 359 - 362: Ανοίγει το παράθυρο της Εικόνα 12, όπου ο Χρήστης έχει τέσσερις επιλογές, (α) Yes, (β) No, (γ) Cancel και (δ) κλείσιμο παραθύρου. Η επιλογή του χρήστη αποθηκεύεται στην ακέραια μεταβλητή result.

Γραμμές 363 – 380: Με μία switch γίνεται η διαχείριση των τεσσάρων επιλογών.

Γραμμές 364 – 374: Η περίπτωση που ο Χρήστης απάντησε Yes και γίνεται αντικατάσταση του αρχείου.

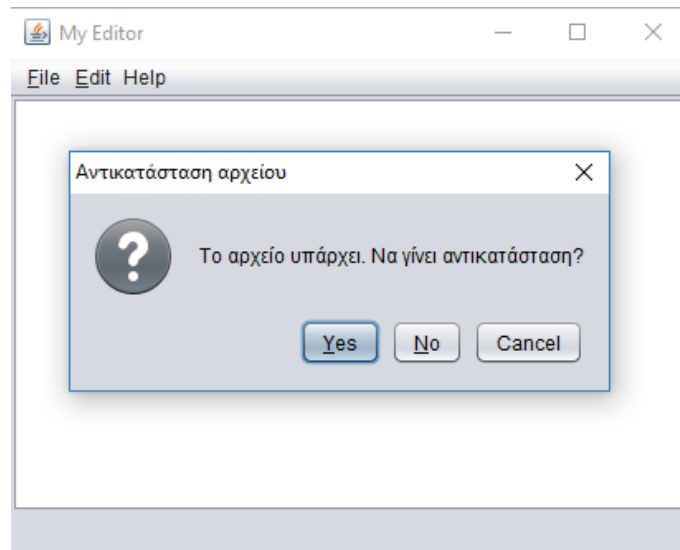
Γραμμές 375 – 376: Η περίπτωση που ο Χρήστης απάντησε No και ξαναεκτελείται το «Save As» από την αρχή.

Γραμμές 377 – 378: Η περίπτωση που ο Χρήστης απάντησε Cancel ή Close και κλείνει το παράθυρο.



```
358 void mysave() {
359     int result = JOptionPane.showConfirmDialog(this,
360         "Το αρχείο υπάρχει. Να γίνει αντικατάσταση?",
361         "Αντικατάσταση αρχείου",
362         JOptionPane.YES_NO_CANCEL_OPTION);
363     switch(result) {
364         case JOptionPane.YES_OPTION:
365             FileWriter fw;
366             try {
367                 fw = new FileWriter(filename);
368                 jTextArea1.write(fw);
369                 fw.close();
370                 setTitle(filename);
371             } catch(IOException exc) {
372                 myerrorMessage(exc.getMessage());
373             }
374             return;
375         case JOptionPane.NO_OPTION:
376             SaveAsFile.doClick();
377         case JOptionPane.CLOSED_OPTION:
378         case JOptionPane.CANCEL_OPTION:
379     }
380 }
```

Εικόνα 11: Ο κώδικας διαχείρισης της αντικατάστασης αρχείου



Εικόνα 12: Το παράθυρο αντικατάστασης αρχείου